SEMI-ANNUAL PROGRESS REPORT
OFFICE OF NAVAL RESEARCH

August 1, 1997

ONR Grant No: N00014-95-1-0790

Scientific Officer: Helen Gigley

Principal Investigators

John Stasko (PI)
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
stasko@cc.gatech.edu

Mark Guzdial (Co-PI)
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
guzdial@cc.gatech.edu

Richard Catrambone (Co-PI)
School of Psychology
Georgia Institute of Technology
Atlanta, GA 30332-0170
rc7@prism.gatech.edu

Ashwin Ram (Co-PI)
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
ashwin@cc.gatech.edu

Our project focuses on researching and defining architecture guidelines for developing multimedia systems to support human learning. We base these guidelines on principles from cognitive science. During the last six months our efforts have been focused on two fronts. The first has been the continued development of an environment for teaching students chemistry, specifically, Lewis Structures. The second has been the specification of a generalized architecture for cognitive multimedia learning environments that operate on a web-based system. We are interested in how presentation factors and orientation tasks affect learning.

November 16, 1996 - May 16, 1997

Important Findings and Problems Encountered

ChemLab

We conducted two new experiments using the ChemLab software. The first of the new experiments was conducted using students at Emory University. We made a number of usability changes to the ChemLab software that we had used in a prior experiment based on feedback from students.

In the prior experiment we had used two conditions: directed browsing and problem solving. In the directed browsing condition students were asked a series of fact questions concerning Lewis Structures (e.g., How many valence electrons does Sulphur have? What is a lone pair? What is the most electronegative element? What is the chemical formula for Hydrazine?) and could navigate ChemLab to find answers. In the problem solving condition students were asked to solve a few Lewis Structure problems, that is, to determine the distribution of electron pairs and bonds for a particular molecule. After exploring ChemLab in one of these two ways, students were given a post-test (they were warned ahead of time they would be getting a post-test after interacting with the software). We did not find any difference in post-test performance between the two groups. We did, however, find differences in their browsing patterns in ChemLab suggesting that the orienting tasks (answering questions or solving problems) did influence how they interacted with

the software. One concern though was that we did not have a control group that could serve as a baseline to compare the performance of the other two groups.

In the first of the new experiments we had four conditions: directed browsing, problem solving, paper, and no ChemLab ($N = 124$). The first two conditions were the same as in the prior study (albeit with an improved interface). The paper condition involved students receiving a print-out of the Hypercard stack for ChemLab and being asked to study it in preparation for the post-test. Finally, students from the class who did not interact with ChemLab served as a "true" baseline. All students from the class took a final exam as part of their regular chemistry course. We were given permission to obtain final exam scores as well as copies of the final exams (coded by experimentally-assigned numbers). This allowed us to examine in detail performance on final exam questions dealing specifically with Lewis Structure problems.

The results from this study suggest that the ChemLab software did not help performance. However, once again their may be an interface-design explanation for this. First, consider the results. Performance on the post-test and the final exam by the different groups was as follows:

	Paper	Directed Browsing	Problem Solving	No ChemLab
Avg. No. of Post-Test Fact Questions Cor. (max = 7)	5.9	5.3	4.6	N/A
Avg. No. of Post-Test Problems Cor. (max = 4)	1.8	0.8	1.0	N/A
Avg. No. of Lewis-Structure Related Final Exam Problems Cor. (max = 4)	3.0	3.5	3.0	3.1

The performance differences on the post-test are statistically significant and suggest that the paper group was generally outperforming the other two on the post-test. The differences among the groups on the final exam were not significant although there was a trend favoring the directed browsing group.

Post-experiment interviews with a subset of students who used ChemLab suggested that one problem with the instructions in the software was that they did not make it clear that once students finished answering the orienting questions or problems the program would terminate and they would then go directly to the post-test. As a result, many of the students who interacted with the ChemLab software (as opposed to the paper version) spent relatively little time with the material. This may have accounted for the superior performance of the paper group on the post-test.

Performance on the final exam (which occurred several weeks after the experiment) suggests that the use of ChemLab might provide better retention of material as indicated by the trend towards better performance by the directed browsing condition.

In the second of the new experiments we altered the instructions to the ChemLab software to make it clearer to students that once they were done answering questions the ChemLab software would terminate. We ran this experiment at Georgia Tech partly in order to see if overall performance would be similar to our Emory sample. We were able to get only about seven participants per condition unfortunately and were not allowed access to final exams. While overall performance was higher than the Emory sample, the relative performance among the groups was about the same with the paper group showing some advantages over the other groups:

	Paper	Directed Browsing	Problem Solving
Avg. No. of Post-Test Fact Questions Cor. (max = 7)	6.3	5.6	5.4
Avg. No. of Post-Test Problems Cor. (max = 4)	2.7	2.4	2.3

We have identified several changes we would like to make to the ChemLab software to better support navigating the system. These changes should also avoid the problem of students unknowingly finishing the program prematurely and having the program terminate before they have finished looking at all of the information.

One area of change will involve the introduction. Currently the introduction is approximately 10 screens of information, much of it involving how to use the system itself. This may have caused many students to become bored and inattentive before beginning the main part of the program. It is expected that by reworking some other aspects of the ChemLab interface, and by revising and reordering the introduction, the introduction can be shortened to 4-6 screens of information, most of it involving necessary background information about bonds and bond formation. In addition, the program will not terminate when learners are done answering questions/problems. Rather, it will terminate when learners indicate they are done exploring. Finally, greater emphasis will be placed on what the overall activity flow will be like, namely moving between the problem/question screen, the information screen, and the molecule construction kit.

Several other proposed changes to the ChemLab interface itself involve the method of navigation and the representation of the visual affordances for navigation. For instance, the full flow chart map of the entire sequence of steps will be made "clickable," so that students may click on the box for a particular step and be taken directly to the information for that step (currently the full map is not clickable - it only shows a full view of the steps to be taken). This will provide a more direct method of navigating from a global view of the overall procedure to the information details for a particular step in the procedure. Another change will be to make the location of the current step on the information screen more stable and informative. Currently the position of the boxes representing the previous, current, and next steps on the information window move around, and thus do not provide any information about whether they are primary steps, primary sub-steps, or secondary sub-steps. We believe that this may have made it more difficult to understand how the current step fit into the overall procedure. Also, the "Back" and "Next" buttons will be removed, and arrows pointing up and down in the navigation area of the information screen will be added that will scroll the steps displayed in that area up and down. Students will have to click directly on a step to display the information for that step (removing some of the ambiguity of "Back" and "Next"). The "Home" button will be reworded to say "Back to Step 1" to be more direct. The information radio buttons (definition, example, worked problem, and problem set) will be moved toward the top of the window to emphasize them more. The "Why?" button will be moved over to the right side of the window with the actual information content and away from the navigation buttons. Finally, the remaining navigation buttons (Full Map, Work, MCK, Back to Step 1) will be consolidated and placed in between the radio buttons and the clickable mini-navigation map.

Overall this reorganization should make the intended flow and actions more intuitive and easier to perform for the user, and clearly distinguish between the navigation and information functions of the various window elements.

Creating a Generalized Architecture for Cognitive Multimedia Learning Environments

Based on our experiences with ChemLab and Algonet, we are continuing to work toward a generalizable architecture for cognitive multimedia learning environments. We noted that our efforts to test alternative navigation and media selections were hampered by the complexity of building testbeds. While still exploring these issues in ChemLab, we have chosen to develop a generalized architecture in which we could easily change navigation, content, and even domain.

As reported in past reports, we have developed a knowledge structure based on cognitive multimedia theory that we hypothesize can be used to support students in learning how to conduct procedures (from repair to troubleshooting to installation) in any domain, using a wide variety of approaches (from direct instruction to scaffolded apprenticeship, from menu-based navigation to virtual reality, and from text-only to full-video multimedia content). Our first step toward testing our generalizability hypothesis is to construct a simple educational application structured with this architecture. The example domain that we have chosen is "plumbing," where our content includes how to clear a clogged drain, how to install a toilet, and how the hot water system works at an abstract level.

We implemented the core architecture (in Java) that would display content based on student experience and a set of display rules (in a production rule format); basically, the core is a cognitive multimedia display engine. The idea is that content would be provided to the display engine along with a set of rules for how to present this content and a memory of information that the user had already provided, e.g., "If the user has identified the problem as a leaky faucet, present the known causes for a leaky faucet along with links to the procedures for testing for each of those causes." Next, we needed content and a sample set of rules.

We developed a simple database application (in Apple Hypercard) for constructing knowledge elements to represent this content. However, after constructing several pieces of the knowledge-base, we found that the database application was clumsy to use and difficult to express complex content in. Basically, our problem was that we were constructing knowledge elements that were relating to other elements in a variety of ways, and that was difficult to represent in a simple user interface.

Therefore, we decided to design and develop a markup language which would allow us to represent linkages between knowledge elements at a symbolic level. We are referring to this language as PML (Procedural Markup Language). We have successfully:

- Defined PML (using the SGML standard DTD [document type definition] structure; see below),
- Written a simple PML to HTML translator to provide a means for reviewing the knowledge structures,
- Translated our database content into PML, and
- Translated our PML content into HTML for review.

All of these components are available for inspection at:
`http://www.cc.gatech.edu/gvu/cognitive/cognitive-multimedia/home.html`

In parallel, we developed a first set of rules for how to present instruction from our knowledge-base. These rules are used in the current PML-to-HTML translator, but in a static form — i.e., the static HTML cannot remember previous student selections. Our next step is to use the PML content to populate our cognitive multimedia engine and to encode our rules as production rules usable by the multimedia engine. At that point, we will have completed a proof-of-concept of the

generalizable system and can move on to conduct further experiments on navigation and media choices and push on to explore other domains.

```
<!DOCTYPE PML.dtd [

This is the document type description for a language we're calling
PML.  First we'll describe the elements in the language.

<!ELEMENT PML      - - (thing | state | procedure)+ >

A PML document consists of "thing"s, "state"s and "procedure"s.  There
may be any number of each of them and they can occur in any order but
there must be at least one element in the document.  A PML document may
not be empty.

<!ELEMENT (thing | state | procedure) - - (name+, author+, (description+ &
justification? & relations? & example* & counter* )) >

Things, states, and procedures consist of the following: at least one
name followed by at least one author.  Then the following may occur in
any order: at least one description, only one justification, only one
set of relations, any number of examples, any number of counterexamples.
"Any number" includes zero.  When something occurs more than once, the
occurances must be "grouped" together - you can't have examples mingled
with descriptions, for instance.  Procedures may no longer be nested.

<!ELEMENT (name | author | description | justification | example | counter)  -
- (#PCDATA) >

Names, authors, descriptions, etc. may occur inline and are composed of
character data only -- they may not include other tags defined in PML.

<!ELEMENT relations - - (link+) >

Relations consists of a set of links.  There must be at least one.

<!ELEMENT link      - o (target+)>

A link consists of a set of targets.  There must be at least one.
Link does not require an end tag because it can only be followed
by another link or the end of the relations tag.

<!ELEMENT target    - o EMPTY>

A target is an empty tag and therefore does not require an end
tag.  An attribute (defined below) will be used to refer to the
element we're using as the target.  Doing it this way allows the
parser to flag when we're refering to an element that doesn't
exist.  Here's what a sample of the relations part would look like:

<RELATIONS>
<LINK type=steps>
  #The ordering here is important and implies Next & Previous links
  <TARGET tid="Project 2, Step 1">  #These are the IDs of other procedures
  <TARGET tid="Project 2, Step 2">
  <TARGET tid="Project 2, Step 3">
  <TARGET tid="Project 2, Step 4">
```

```
<LINK type=relatedto>
  <TARGET tid="Another procedure">
  <TARGET tid="Some thing">
<LINK type=isa>
  <TARGET tid="generic Project 2">
</RELATIONS>
```

Now we define the attributes for the various elements.

```
<!ATTLIST (thing | state | procedure)
          id              ID        #REQUIRED>
```

Things, states and procedures are required to have an ID which will be unique across the database.

```
<!ATTLIST (author | description | justification | example | counter )
          src             CDATA     #IMPLIED>
```

The tags listed may have an optional "src" attribute to allow these tags to refer to external resources. These will generally be filenames, but can be any character data.

```
<!ATTLIST prodecure
          difficulty      CDATA     #IMPLIED>
```

Procedures may have a difficulty rating. We're not going to define what those ratings are, though.

```
<!ATTLIST description
          detail          CDATA     #IMPLIED>
```

Descriptions may have a detail rating. Again, we'll leave the actual values undefined.

```
<!ATTLIST link
          type      (uses | isa | hasa | connectsto | relatedto | example |
counterexample | steps | pre | post | problem | repair) #REQUIRED>
```

Links must have a type specified. The valid types are listed above. We could open this up so that new types may be specified as well.

```
<!ATTLIST target
          tid           IDREF     #REQUIRED>
```

Targets are required to have a tid attribute (target-ID). The tid should be an ID which is elsewhere in the system.

```
] >
```

Changes in Overall Plan and Personnel

We replaced one paid graduate student (Viren Shah) with two unpaid grad students (Teresa Hubscher-Younger, Coleen Kehoe) who are interested in the project because it is related to their thesis work.

Experiments

Described above.

Journal Publications

None

Papers Submitted for Publication

None

Papers in Refereed Conference Proceedings

None

Paper Presentations

None

Upcoming Research

We will be conducting a new ChemLab experiment with the redesigned interface and improved instructions. Besides differences in browsing patterns and post-test performance, we will also be interested in observing whether the redesigned ChemLab software helps students with long-term retention as measured by final exam performance.

We will continue to develop PML and apply it to the home repair in order to come up with a web-based system that we can test.